Modelling and Analysing an Identity Federation Protocol: Federated Network Providers Scenario

Maurice ter Beek (FMT, ISTI–CNR, Pisa, Italy)

joint work with

Marinella Petrocchi (IIT–CNR, Pisa, Italy)

Corrado Moiso (Telecom Italia, Torino, Italy)

YR-SOC 2007, Leicester, UK

Outline of the talk

- Setting
- Identity federation protocols
- Telecom Italia's network protocol for identity federation
- Modelling and analysis
  - Analysis approach
  - Crypto-CCS specification language
  - Formalisation of two scenarios of the network protocol
  - Analyses and results of a man-in-the-middle attack
- Conclusions and future work

Setting

- Formal modelling and analysis of security protocols is an active branch of computer security
- Many techniques proved successful (based on process algebras, authentication logic, type systems, etc.)
- We formally specify three user scenarios of a network protocol for identity federation proposed by Telecom Italia, at the same time adding primitives to assure basic security properties
- We then model check our specifications to test their correctness

Identity federation protocols

- Growing interest in defining telecommunication protocols that allow a user to access all services belonging to the same *circle of trust* with a (cross-domain) *single sign-on*
- Process of *identity federation*: federating an entity's identity and allowing access to services without explicitly presenting one's credentials time and again
- *Liberty Alliance*: consortium formed to define processes supporting the federation of identities
- Specifications make use of the XML-based *Security Assertion Markup Language* SAML

Security features

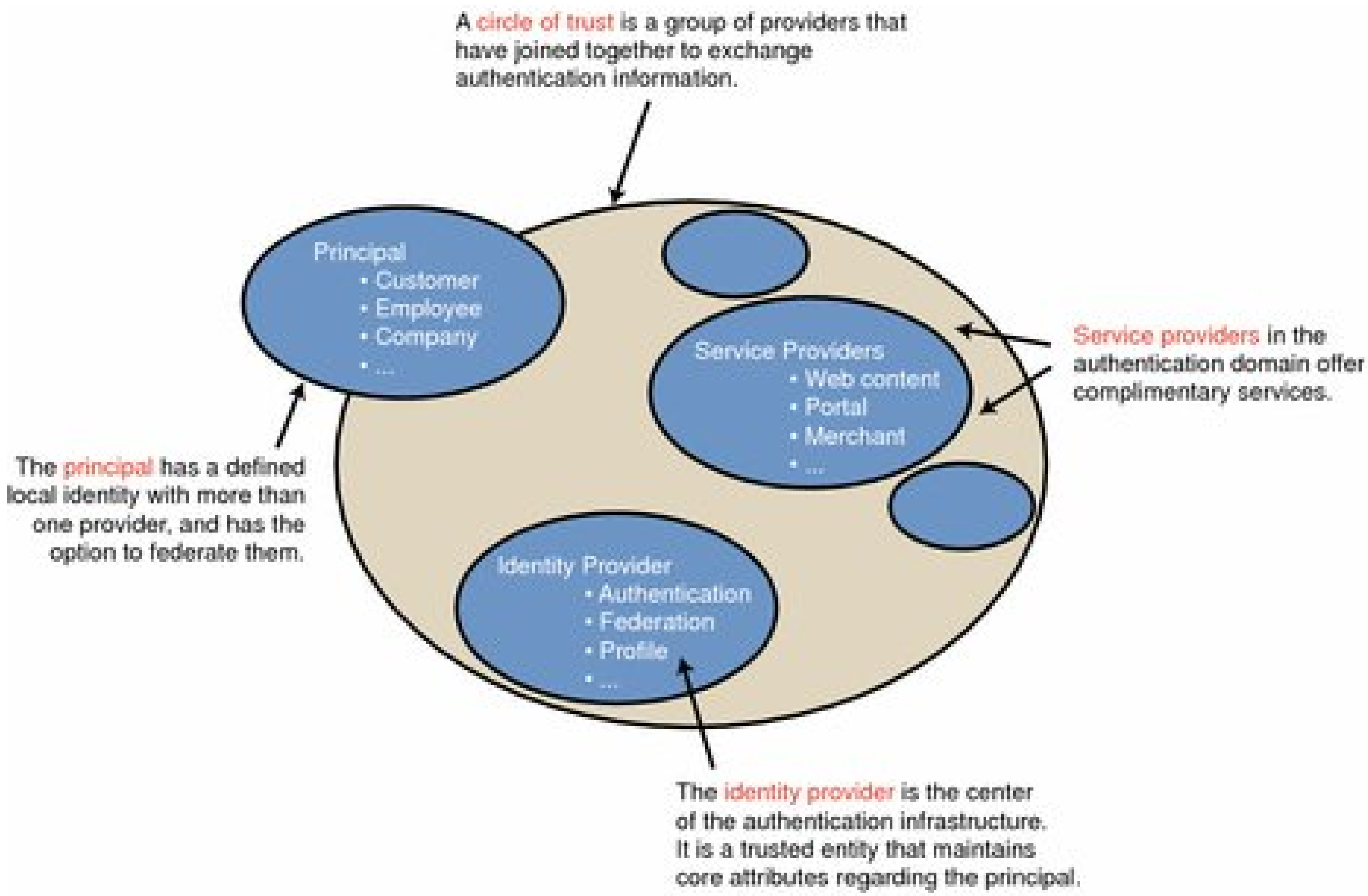
- Limit access to *authenticated* and *authorized* users
- Preserve *privacy* of users:
  - protect sensitive information (e.g. network addresses)
  - guarantee identities without explicitly discovering them
  - only disclose information related to the specific service for which access is requested (e.g. destination preferences if the service is a travel agency)
- (Optional) Grant users *anonymous* access to services (e.g. for temporary federations)

Federating identities example

- ABC airlines and XYZ car rental company decide to create a circle of trust
- Mary has accounts on both ABC's and XYZ's web sites
- She logs into ABC's web site – "*You may share (or federate) your ABC online identity with members of our affinity group, which includes XYZ*"
- Mary likes the idea, so she gives her permission
- Mary goes to XYZ – "*We see you're logged into ABC's web site. Would you like to link your XYZ online identity with your ABC online identity?*" OK!

⇒ In the future, when Mary goes to either ABC's or XYZ's web site, she only needs to log into one to be automatically logged into the other.

Federated identity architecture



Some main features

- Authentication is delegated to an *identity provider*, allowing *single sign-ons*
- A user token is a sequence of characters that identifies the user to each pair of parties in the circle of trust
- User tokens are opaque, i.e. have meaning only for the two parties that federate their users' identities
- Problem: handle identity and authentication information of end users that access services on convergent networks through multiple telecommunication channels (e.g. ADSL, GPRS/UMTS, SMS)

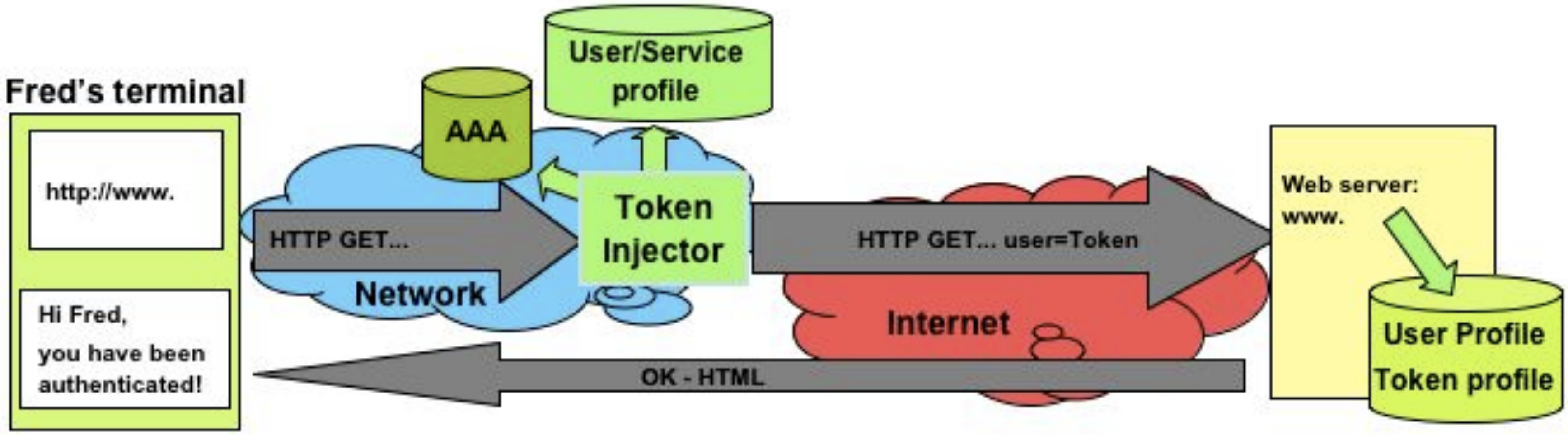
The network protocol

proposed by Telecom Italia @ ICIN'06

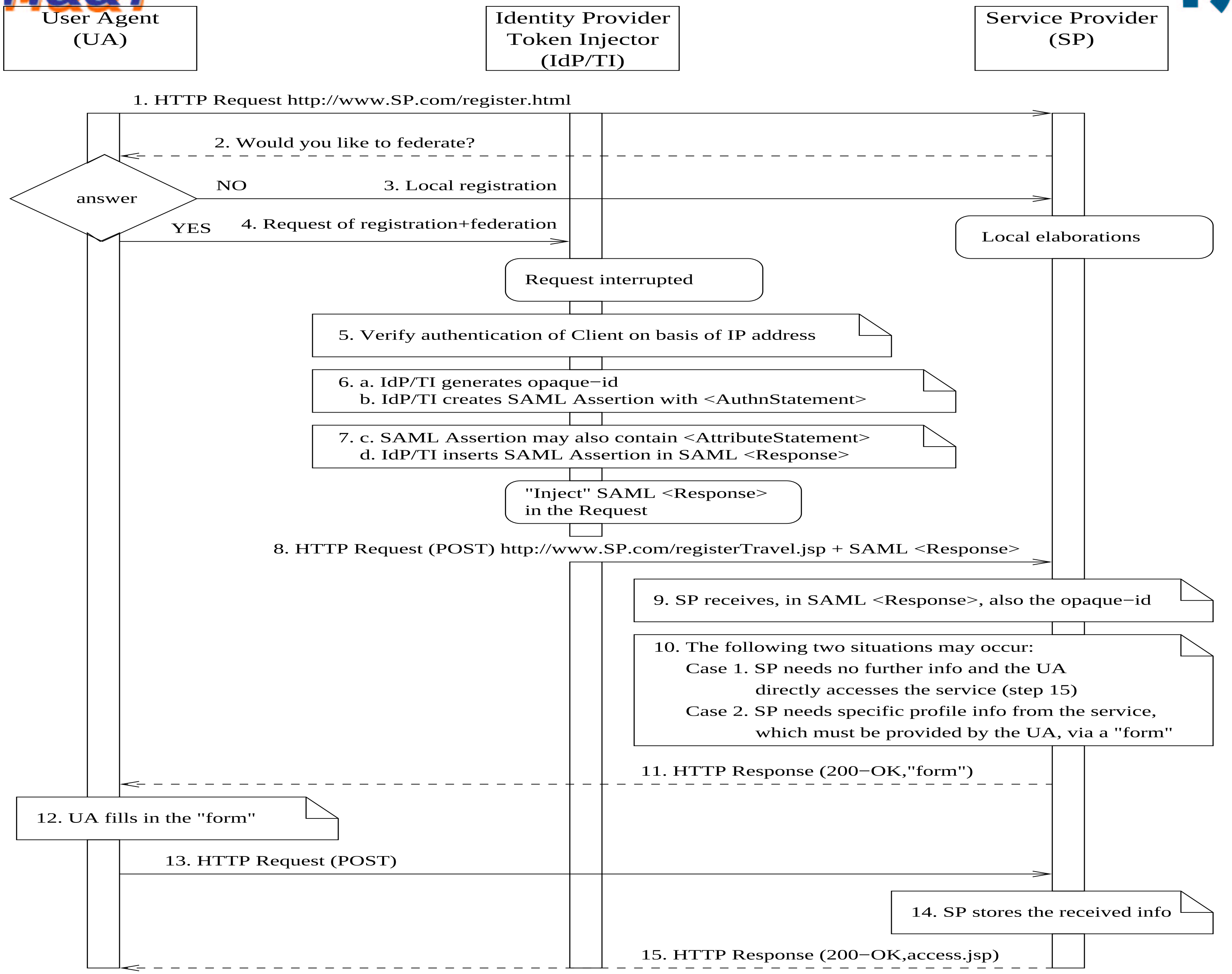
- is an identity federation protocol
- permits users to access services through different access networks (e.g., fixed and mobile)
- gives the network provider the role of identity provider, based on the idea that providers are in a privileged position to pass user information obtained within their security domain to the application level

⇒ Services thus rely on the authentication information provided by the access network

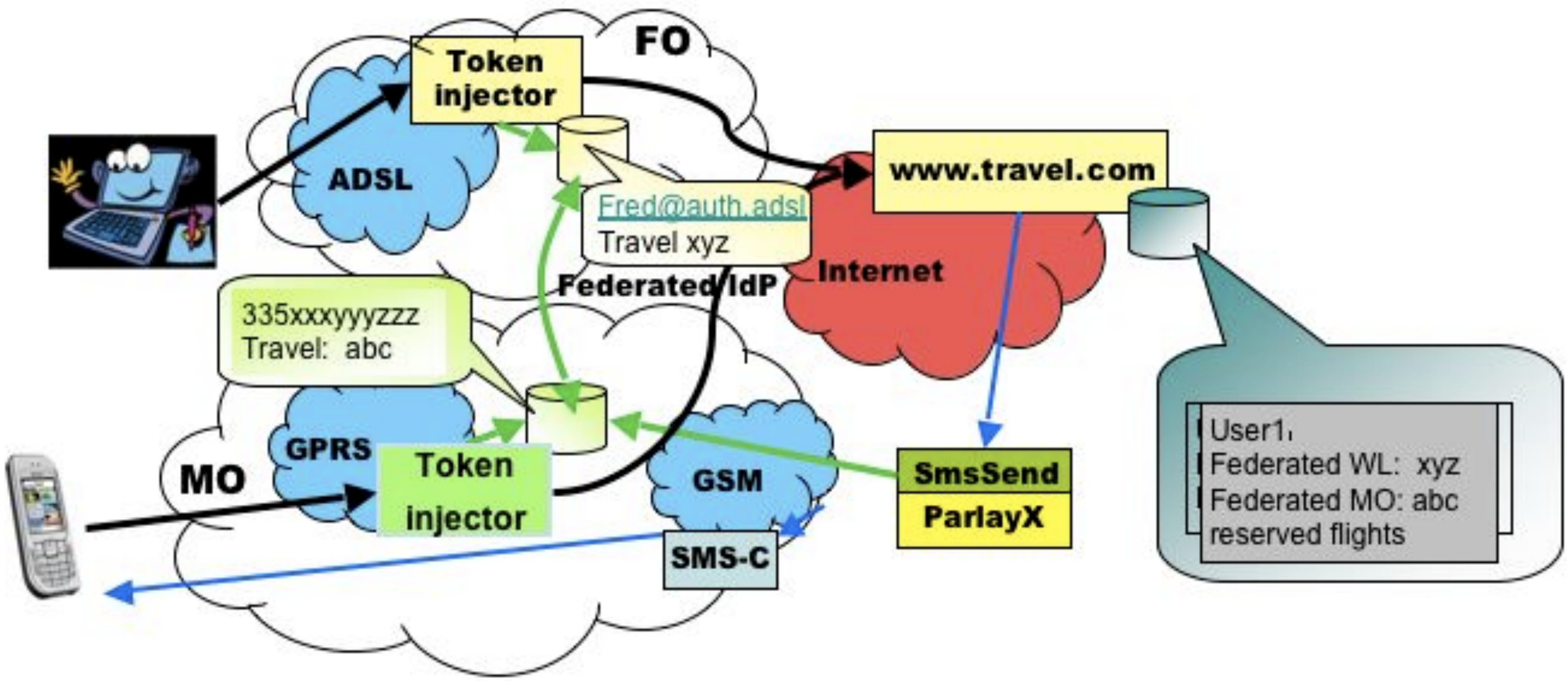
Token injector mechanism

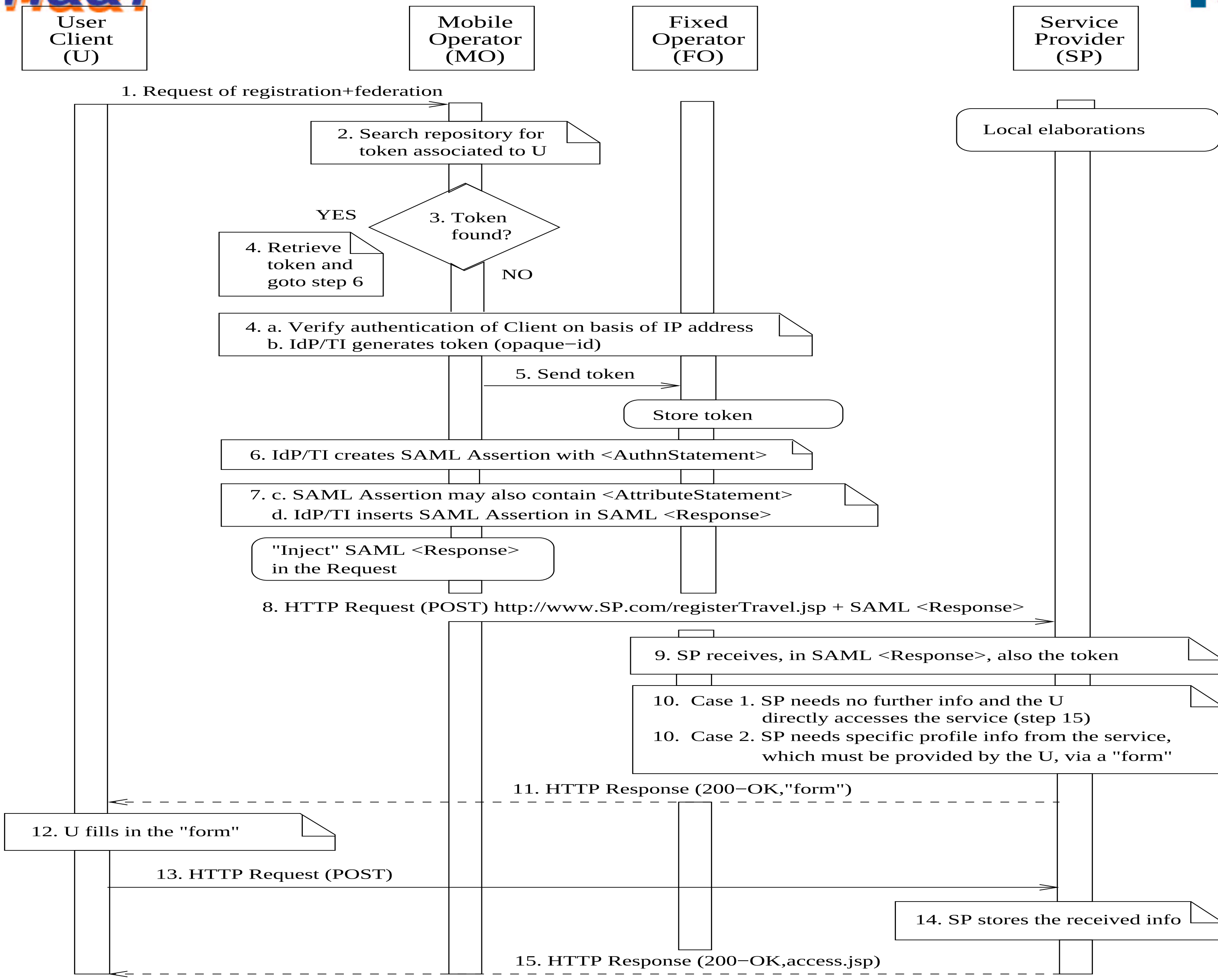


- intercepts HTTP access requests
- (generates) and injects tokens
- forwards them to the applications


User Agent (UA)
Identity Provider Token Injector (IdP/TI)
Service Provider (SP)
1. HTTP Request http://www.SP.com/register.html
2. Would you like to federate?
answer
NO
3. Local registration
YES
4. Request of registration+federation
Local elaborations
Request interrupted
5. Verify authentication of Client on basis of IP address
6. a. IdP/TI generates opaque−id
b. IdP/TI creates SAML Assertion with <AuthnStatement>
7. c. SAML Assertion may also contain <AttributeStatement>
d. IdP/TI inserts SAML Assertion in SAML <Response>
"Inject" SAML <Response> in the Request
8. HTTP Request (POST) http://www.SP.com/registerTravel.jsp + SAML <Response>
9. SP receives, in SAML <Response>, also the opaque−id
10. The following two situations may occur:
Case 1. SP needs no further info and the UA directly accesses the service (step 15)
Case 2. SP needs specific profile info from the service, which must be provided by the UA, via a "form"
11. HTTP Response (200−OK,"form")
12. UA fills in the "form"
13. HTTP Request (POST)
14. SP stores the received info
15. HTTP Response (200−OK,access.jsp)

Multiple access networks




User Client (U)
Mobile Operator (MO)
Fixed Operator (FO)
Service Provider (SP)
1. Request of registration+federation
2. Search repository for token associated to U
Local elaborations
YES
3. Token found?
4. Retrieve token and goto step 6
NO
4. a. Verify authentication of Client on basis of IP address
b. IdP/TI generates token (opaque−id)
5. Send token
Store token
6. IdP/TI creates SAML Assertion with <AuthnStatement>
7. c. SAML Assertion may also contain <AttributeStatement>
d. IdP/TI inserts SAML Assertion in SAML <Response>
"Inject" SAML <Response> in the Request
8. HTTP Request (POST) http://www.SP.com/registerTravel.jsp + SAML <Response>
9. SP receives, in SAML <Response>, also the token
10. Case 1. SP needs no further info and the U directly accesses the service (step 15)
10. Case 2. SP needs specific profile info from the service, which must be provided by the U, via a "form"
11. HTTP Response (200−OK,"form")
12. U fills in the "form"
13. HTTP Request (POST)
14. SP stores the received info
15. HTTP Response (200−OK,access.jsp)

Analysis approach

- We specify the protocol in the process algebra Crypto-CCS, which is CCS plus some cryptographic primitives
- We specify the properties to be verified by logic formulae
- We add a Dolev-Yao-like *intruder* to the specification, whose behaviour is implicitly defined by the semantics of the language
- We verify a property by monitoring the *intruder's knowledge*, which is the set of messages the intruder initially knows plus those received during computation

Crypto-CCS

- Set of processes communicating via message passing
- Inference system models possible operation on messages

$$r = \frac{m_1 \quad \cdots \quad m_n}{m_0}$$

$S := S_1 \parallel S_2 \mid A$	*compound systems*
$A := \mathbf{0} \mid p.A \mid [m_1 \cdots m_n \vdash_r x]A; A_1$	*sequential agents*
$p := c!m \mid c?x$	*prefix constructs*

Informal semantics of Crypto-CCS

$c!m$	send message m over channel c
$c?x$	receive message m over channel c
$\mathbf{0}$	do nothing
$p.A$	perform p and then behave as A
$[m_1 \cdots m_n \vdash_r x]A; A_1$	inference construct
$S_1 \parallel S_2$	parallel composition plus synchronization

Example: $[m \quad pk_y^{-1} \vdash_{sign} x]A; \mathbf{0}$

A process that uses rule $sign$ to obtain a digitally signed message from plaintext message m and private key pk_y^{-1} and then behaves as A, or otherwise does nothing

An example inference system

for public-key cryptography

$$\frac{x \quad y}{\mathit{Pair}(x,y)}\ (\mathit{pair}) \qquad \frac{x \quad pk_y^{-1}}{\{x\}_{pk_y^{-1}}}\ (\mathit{sign}) \qquad \frac{x \quad \mathit{KEY}}{\{x\}_{\mathit{KEY}}}\ (\mathit{enc})$$

$$\frac{\mathit{Pair}(x,y)}{x}\ (\mathit{1st}) \qquad \frac{\{x\}_{\mathit{KEY}} \quad \mathit{KEY}}{x}\ (\mathit{dec})$$

$$\frac{\mathit{Pair}(x,y)}{y}\ (\mathit{2nd}) \qquad \frac{\{x\}_{pk_y^{-1}} \quad pk_y}{x}\ (\mathit{ver}) \qquad \frac{x}{x}\ (\mathit{check})$$

Federated registration

$$\begin{array}{llcl} c_0 & \mathit{U} \mapsto \mathit{IdP} & : & r \\ c_1 & \mathit{IdP} \mapsto \mathit{SP} & : & \{r, \mathit{SAML\ assertion}\}_{K^{-1}_{IdP}} \\ c_2 & \mathit{SP} \mapsto \mathit{U} & : & \{ok/ko\}_{K^{-1}_{SP}} \end{array}$$

1. user U asks identity provider IdP and service provider SP to federate
2. request r intercepted by IdP
   - $\Rightarrow$ authenticate U
   - $\Rightarrow$ generate token id_U
   - $\Rightarrow$ assemble *SAML assertion*
3. SP grants/denies access to U

SAML assertion

A *SAML assertion* declares "*Subj* is authenticated"

$\{$*Subj*, *AuthStat*, *AttrStat*$\}_{KEY}$ (encrypted *SAML assertion*)

Subj token id_U, univocally identifying U

AuthStat authentication statement, asserting U was authenticated (and the mechanism by which)

AttrStat attribute list of U plus nonce n_U^{IdP} to avoid replay attacks

$\{r,$ *SAML assertion*$\}_{K_{IdP}^{-1}}$ (signed by IdP for authenticity)

$SP_0(0) \doteq c_1?x_m.SP_1(x_m)$	*receive SAML assertion + request*
$SP_1(x_m) \doteq [x_m \quad k_{IdP} \vdash_{ver} x_p]$	*verify signature,*
$[x_p \vdash_{2nd} x_{enc}]$	*extract encryption,*
$[x_{enc} \quad KEY \vdash_{dec} x_{dec}]$	*decrypt,*
$[x_{dec} \vdash_{1st} x_{pair}]$	*extract pair: token + AuthStat,*
$[x_{dec} \vdash_{2nd} x_{n_U^{IdP}}]$	*extract nonce,*
$[x_{pair} \vdash_{1st} x_{id_U}]$	*extract token,*
$[x_{pair} \vdash_{2nd} x_{auth}]$	*extract AuthStat,*
$[x_{auth} \vdash_{check} x_{auth}]$	*test correctness AuthStat,*
$[x_{n_U^{IdP}} \vdash_{check} x_{n_U^{IdP}}]$	*test freshness nonce,*
$[x_{id_U} \quad x_{n_U^{IdP}} \vdash_{pair} (x_{id_U}, x_{n_U^{IdP}})]$	*build pair to store,*
$c_S!(x_{id_U}, x_{n_U^{IdP}})$	*store token + nonce pair,*
$[access \quad k_{SP}^{-1} \vdash_{sign} x_{sign}]$	*prepare signature to*
$c_2!x_{sign}.\mathbf{0}$	*grant access and stop*

Federated network providers

c_{MF} *FO* $\leftrightarrow$ *MO* assumed secure: share secret key KEY_{FM}

c_0	*U* $\mapsto$ *MO*	:	r
c_{MF}	*MO* $\mapsto$ *FO*	:	$\{id_U, U\}_{KEY_{FM}}$
c_1	*MO* $\mapsto$ *SP*	:	$\{r, \textit{SAML assertion}\}_{K^{-1}_{MO}}$
c_2	*SP* $\mapsto$ *U*	:	$\{ok/ko\}_{K^{-1}_{SP}}$

We slightly enrich network protocol presented @ ICIN'06:

When *FO*/*MO* receives r from U, search repository for id_U

- If found, then retrieve it and continue as usual
- Else, generate id_U and send it to federated provider, where stored for other interactions between *U* and *SP*

Crypto-CCS specification – MO

$MO_0(0, n_U^{MO}, id_U, KEY_{FM}) \doteq$

$c_0?x_r.MO_1(x_r, n_U^{MO}, id_U, KEY_{FM})$ *receive request*

$MO_1(x_r, n_U^{MO}, id_U, KEY_{FM}) \doteq [id_U \quad U \vdash_{pair} (id_U, U)]$ *create pair,*

$[(id_U, U) \quad KEY_{FM} \vdash_{enc} \{(id_U, U)\}_{KEY_{FM}}]$ *encrypt pair,*

$c_{MF}!\{(id_U, U)\}_{KEY_{FM}}.$ *send token to FO,*

$[id_U \quad auth \vdash_{pair} (id_U, auth)]$ *create pair,*

$[(id_U, auth) \quad n_U^{MO} \vdash_{pair} ((id_U, auth), n_U^{MO})]$ *create pair,*

$[((id_U, auth), n_U^{MO}) \quad KEY \vdash_{enc}$
$\{((id_U, auth), n_U^{MO})\}_{KEY}]$ *encrypt pair,*

$[x_r \ \{((id_U, auth), n_U^{MO})\}_{KEY} \vdash_{pair}$
$(x_r, \{((id_U, auth), n_U^{MO})\}_{KEY})]$ *create pair,*

$[(x_r, \{((id_U, auth), n_U^{MO})\}_{KEY}) \quad k_{MO}^{-1} \vdash_{sign} x_{sign}]$ *sign pair,*

$c_1!x_{sign}.\mathbf{0}$ *send SAML assertion + request and stop*

A man-in-the-middle attack

Can intruder *X* intercept (modify) a conversation between MO and SP, without the latter being aware of this?

PROPERTY

"whenever SP concludes the protocol apparently with MO, it was indeed the latter that executed the protocol"

Use two special actions in our Crypto-CCS specification:

- *commit*(*a,b*): a indeed finished the protocol with b
- *run*(*b,a*): a indeed started the protocol with b

Property

Does a computation exists such that:

- *SP* is convinced to have finished talking with *MO*, while in reality *MO* never started talking with *SP*
- *FO* is convinced to have finished talking with *MO*, while in reality *MO* never started talking with *FO*

$$(\mathit{commit}(SP,MO) \text{ AND } (\text{NOT } \mathit{run}(MO,SP)))$$
$$\text{OR}$$
$$(\mathit{commit}(FO,MO) \text{ AND } (\text{NOT } \mathit{run}(MO,FO)))$$

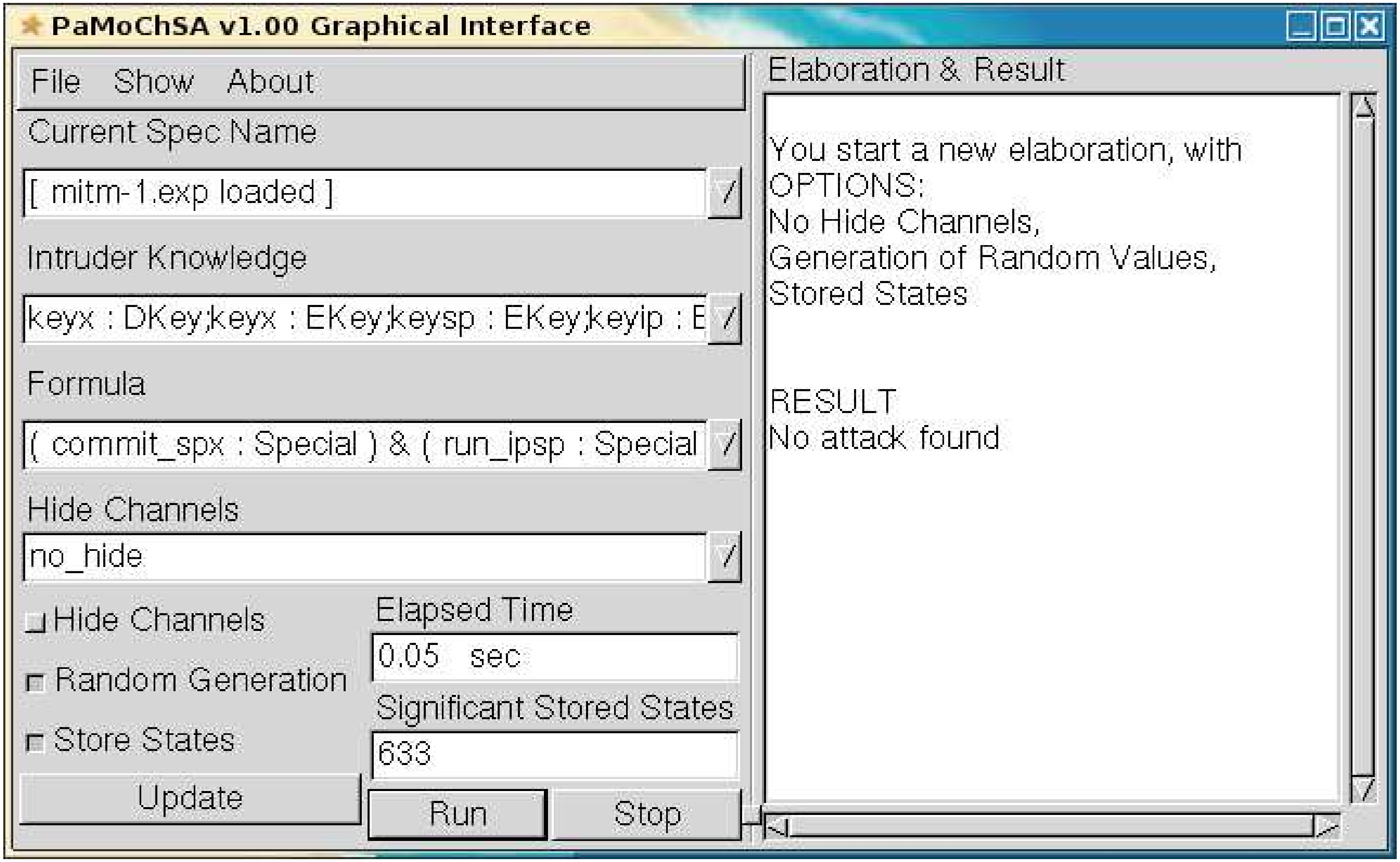
Input model checker

PaMoChSA v1.0 developed at IIT–CNR

- Specification file: `mitm-2.exp`
- Logic formula: (*commit*(*SP,MO*) AND (NOT *run*(*MO,SP*))) OR (*commit*(*FO,MO*) AND (NOT *run*(*MO,FO*)))
- Initial knowledge: $\{pk_X, pk_X^{-1}, pk_{MO}, pk_{FO}, pk_{SP}\}$
- Result: **No attack found**

(analogously for federated registration)

PaMoChSA's graphical interface



Conclusions

- We advocate the use of formal methods in the design phase of protocols so as to *obtain well-defined protocols guaranteed to satisfy certain desirable properties*
- The results of our initial analyses strengthen our confidence in our formal specifications
- In particular, these results lead us to believe that we correctly inserted digital signatures, encryption and nonces into the network protocol as originally proposed by Telecom Italia

Future work

- Extend our analyses by considering:
  - more user scenarios
  - more security issues (e.g. unsubscription & anonymity)
- Presented paper at AICT'07 (*3rd Advanced International Conference on Telecommunications*, IEEE Computer Society) that covers the *federated registration* scenario
- Deal with quantitative extensions of formal methods and tools (such as probabilistic specification languages and stochastic model checkers)